An extended abstract of this work appears in *Proceedings of the 5th International Conference (SCN '06)*
(6 – 8 september 2006, Maiori, Italy)
M. Yung Ed., Springer-Verlag, LNCS 4116, Pages 156–172.

About the Security of MTI/C0 and MQV

Sébastien Kunz-Jacques[1,2] and David Pointcheval[1]

[1] École normale supérieure, 45 rue d'Ulm, 75005 Paris, France
David.Pointcheval@ens.fr
[2] DCSSI Crypto Lab, 51 boulevard de La Tour-Maubourg
F-75700 Paris 07 SP, France
kunzjacq@yahoo.fr

Abstract. The main application of cryptography is the establishment of secure channels. The most classical way to achieve this goal is definitely the use of variants of the signed Diffie-Hellman protocol. It applies a signature algorithm on the flows of the basic Diffie-Hellman key exchange, in order to achieve authentication. However, signature-less authenticated key exchange have numerous advantages, and namely from the efficiency point of view. They are thus well-suited for some constrained environments. On the other hand, this efficiency comes at the cost of some uncertainty about the actual security.
This paper focuses on the two most famous signature-less authenticated key exchange protocols, MTI/C0 and MQV. While the formal security of MTI/C0 has never been studied, results for the plain MQV protocol are still debated. We point out algorithmic assumptions on which some security proofs can be built in the random oracle model. The stress is put on implementation aspects that must be properly dealt with in order to obtain the expected security.
Some formalizations about authenticated key exchange, and the generic model, are of independent interest.

Key words: Key Exchange, MTI, MQV, Diffie-Hellman, Security Proof.

1 Introduction

Since the introduction of the Diffie-Hellman protocol in the seminal paper [13], key exchange has played a prominent role in public-key cryptography. It provides two entities communicating on an insecure channel with a common secret value, which can thereafter be used to setup a secure channel. The plain Diffie-Hellman protocol does not provide entity authentication and is therefore vulnerable to "man-in-the-middle" attacks. A classical way to overcome this weakness is to authenticate the flows with strong authentication mechanisms, such as message authentication codes (MAC) or signature schemes (as for instance in the Station-To-Station protocol [14]).

A few proposals apply weaker authentication techniques, which are specific to the key agreement method. Whereas they are signature-less, they provide both strong authentication (the so-called "mutual authentication") and strong secrecy (the so-called "forward-secrecy"). Furthermore, since no signature computations/verifications are needed, they are quite efficient.

This paper focuses on some of these "signature-less protocols". The most well-known algorithms in that category are the MTI family [19, 21] and MQV [20, 27]. More specifically, among the MTI family, we focus on MTI/C0, which is the only variant of the MTI family that can be expected to provide the forward-secrecy. MQV was proposed as a solution to overcome some security weaknesses of MTI/C0. However, one can remark that attacks against the "basic MTI/C0" protocol can be easily prevented when proper and classical safeguards (eg. key confirmation rounds) are added.

As a conclusion, we show that when properly set-up, that is, in a suitably chosen group and with a proper key derivation mechanism, both MTI/C0 and MQV are

secure authenticated key exchange protocols, and even achieve forward-secrecy. We focus on the 3-pass variants of these protocols because no two-pass protocol achieves mutual authentication: the first message can always be replayed by an active attacker. Some two-pass protocols are analyzed in [15, 18].

Related work. Key exchange is closely related to authentication, as illustrated by the "man-in-the-middle attacks". A very general model for these two problems was introduced by Bellare and Rogaway [7]. Bellare, Canetti and Krawczyk [3] followed a different path, by providing a general tool to transform a protocol secure when communications are authenticated into a new protocol secure against an active adversary (able to alter messages). Among other applications, this framework can be applied to the "authenticated key exchange" (AKE) problem.

An extensive comparison of the security properties of some signature-less AKE protocols can be found in [9]; however, no security proof is provided. On the other hand, [8] provides security analyzes of several authenticated variants of the Diffie-Hellman scheme. For such studies, a formal security model is required. We thus review the strongest one, based on the seminal work of Bellare and Rogaway [7], and various extensions from [1, 2, 8].

The security of MQV was recently analyzed and a "hashed" variant, HMQV, was proved [17]. We focus on the plain MQV protocol, and show that proper key derivation is enough to overcome its security weaknesses, like the Unknown Key Share attack of Kaliski [16]. As for MTI/C0, no formal security result was available to our knowledge.

Security Model. Informally, we want to model resistance of a key exchange protocol against active and adaptive attackers. The required security properties are:

- **Semantic security.** If an execution of the protocol successfully terminates between a user A and its intended correspondent B, no one but A and B should possess any information about the key agreed upon;
- **Mutual authentication.** A user A engaged in a key exchange session accepts (actually gets a session key) with B only if it is indeed speaking to B;
- **Forward secrecy.** The disclosure of some user's private keys does not compromise (the semantic security of) previously negotiated keys.

By "active and adaptive attackers", as in [7], we mean that the attacker E has entire control of the communication network, and thus controls all flows between users. Therefore, there is no canonical definition of the partner of some user that runs the protocol. Partnership is defined with the help of views of the exchanged messages between two users. Since we consider forward-secrecy, E is also allowed to (adaptively) corrupt users, which then provide her with their long-term private keys.

More formally, the attacker plays a real-or-random game with a simulator, in which it succeeds if it distinguishes between true negotiated keys and random values. This game models the semantic security (and even the forward-secrecy, if the corruption of players is allowed). Strictly speaking, not mutual authentication, but only *implicit authentication* is guaranteed: when A negotiates a key with B, only A and B can compute the key, however from the point of view of A, there is no guarantee that B did compute the key or even that B was involved in the exchange at all. Key confirmation rounds are however well-known to enhance semantic security into mutual authentication [5, 11].

Note that the classical definition of the semantic security involves a find-then-guess game [7, 4]. In this paper, we use a real-or-random game, which is both stronger [2] and simpler to handle.

Contributions. Proofs are performed in the random oracle model [6] and rely on custom variants of the Diffie-Hellman problem: f-RCDH for MQV and 2-3-CDH for MTI/C0. f-RCDH is a rather non-standard problem, and might well be weaker than plain CDH; however we show that the f-RCDH intractability hypothesis is **equivalent** to the semantic security of MQV, which gives a strong motivation to introduce this new algorithmic problem (while the reduction of f-RCDH to MQV is performed in the random oracle model, the reduction of MQV to f-RCDH is in the standard model) . On the other hand, 2-3-CDH is a rather natural extension of CDH, but we only show that 2-3-CDH intractability hypothesis is at least as strong as the semantic security of MTI/C0.

Since new assumptions are always questionable, besides the security analysis, a large part of the paper is devoted to study the two new problems 2-3-CDH and f-RCDH. In particular, we build on generic group results to provide arguments towards the actual hardness of both problems: they are hard in the generic sense. Moreover, f-RCDH is shown to be equivalent to the classical CDH, under the additional assumption that the truncation function f used in MQV can be modeled as a random oracle. This motivates the replacement of this function of MQV by a proper hash function, as performed in [17].

For this analysis, we construct a simple and new tool of independent interest that allows one to check whether a particular variant of the Diffie-Hellman problem is hard in the generic sense or not.

Organization. The paper is organized as follows. Section 2 introduces a common framework for signature-less authenticated key exchange protocols. Many different protocols, among which MTI variants and MQV, can be plugged into that framework. MTI/C0 and MQV are presented in section 3, together with the corresponding algorithmic hypotheses, 2-3-CDH and f-RCDH. A sketch of the security proof is presented in section 4, while the full proof is postponed to appendix B. Next, in section 5, the new algorithmic assumptions are analyzed. Finally, we sum up in section 6 the key design choices that help make a signature-less key exchange protocol secure. The security model, which is the classical one, is reviewed in appendix A.

2 A Framework for Signature-Less Authenticated Key Exchange

We describe a general framework, in order to deal with signature-less authenticated key exchange protocols. Users are assumed to own public/private key pairs, and the public keys are supposed to be authentic and known to any party of the system.

First, we need some description of the view that a user (A or B) has of the messages exchanged during a session, since this will define the partnership relation.

Session Flow, Partners. We denote by $\mathsf{Flow}(\mathsf{U}, i)$ the bit-string encoding the messages seen by user $\mathsf{U} \in \{\mathsf{A}, \mathsf{B}\}$ during session i, *up to the key material agreement*. It is assumed that

$$\mathsf{Flow}(\mathsf{A}, i) = \mathsf{Flow}(\mathsf{B}, i) \iff \left\{ \begin{array}{l} \text{no message between A and B was} \\ \text{altered in any way during session } i \end{array} \right\}$$

A and B are said to be *partners* in a session i if $\mathsf{Flow}(\mathsf{A}, i) = \mathsf{Flow}(\mathsf{B}, i)$. Informally, if A and B are partners in session i, they share the same key at the end of the session, and the converse should hold except with negligible probability.

Key Material Agreement. Let us now describe a key agreement between two users A and B. In a preliminary phase, one of the users asks the other party to initiate a key negotiation. From the cryptographic standpoint, the only interest of this phase is that the messages exchanged ends up in the session flow like the rest of the exchange: as a consequence, we can assume that the identities of A and B are contained in the session flows.

This phase of the protocol allows A and B to agree on common secret key material from which both the session key and key confirmations are derived:

- A chooses at random an element r_{A} in some space $\mathcal{R}$. Some function φ of r_{A} is sent to B. The function φ might additionally take as input A's private/public key, and B's public key. We name M_{A} all this long term key material available to A.
- B performs the same operation towards A.
- A and B both derive some *key material* KM through another operation ψ satisfying[1] a kind of commutativity property

$$\mathsf{KM} = \psi(\varphi(r_{\mathsf{A}}, M_{\mathsf{A}}), r_{\mathsf{B}}, M_{\mathsf{B}}) = \psi(\varphi(r_{\mathsf{B}}, M_{\mathsf{B}}), r_{\mathsf{A}}, M_{\mathsf{A}}).$$

Key Confirmation. When a user $\mathsf{U} \in \{\mathsf{A}, \mathsf{B}\}$ has computed KM, it can compute the common key $K \in \mathcal{K}$ and the *key confirmations* $\mathsf{KC}(\mathsf{U}')$ for any partners U' (and himself) by

$$K = H(\mathsf{KM} \| 0 \| \mathsf{Flow}(\mathsf{U})) \quad \mathsf{KC}(\mathsf{U}') = H(\mathsf{KM} \| 1 \| \mathsf{ID}_{\mathsf{U}'} \| \mathsf{Flow}(\mathsf{U}))).$$

In this relation, the flows consist of the messages up to and including the exchange of random elements, H is a h-bit hash function (assumed to behave like a random oracle). Both A and B can compute the two key confirmations $\mathsf{KC}(\mathsf{A})$ and $\mathsf{KC}(\mathsf{B})$. But A sends $\mathsf{KC}(\mathsf{A})$ to B, while B sends $\mathsf{KC}(\mathsf{B})$ to A. Each user checks the value sent by the other and rejects the key if this value is incorrect.

3 Formalization of MTI/C0 and MQV

For both MTI/C0 and MQV, G is a cyclic group of prime order p, and g is a generator of G. All random elements are drawn uniformly in the sets mentioned.

3.1 MTI/C0

The private key s_u of a user U is a random element in $\mathbb{Z}_p^\star$ and the corresponding public key equals $K_u = g^{s_u}$.

[1] ψ might reject some values of its first input: the relation holds only when neither $\varphi(r_{\mathsf{A}}, M_{\mathsf{A}})$ nor $\varphi(r_{\mathsf{B}}, M_{\mathsf{B}})$ is rejected

Key Material Agreement. A (resp. B) draws a random element r_a (resp. r_b) in $\mathbb{Z}_p^\star$. A then computes $R_a = {K_b}^{r_a}$ and sends it to B, while A computes $R_b = {K_a}^{r_b}$ and sends it to A. The key material KM is then computed by each user according to the relation

$$\mathsf{KM} = g^{r_a r_b} = {R_a}^{r_b/s_b} = {R_b}^{r_a/s_a}$$

Note that if one of the received values (R_a or R_b) is equal to 1, the recipient aborts the protocol. Thus, using the framework of section 2, we have

$$\varphi(r_a, M_a = (s_a, K_a, K_b)) = R_a = K_b^{r_a}$$

and

$$\psi(R_a, r_b, M_b = (s_b, K_a, K_b)) = \begin{cases} \texttt{abort} & \text{if } R_a = 1 \\ R_a^{r_b/s_b} & \text{otherwise} \end{cases}$$

2-out-of-3 Computational Diffie-Hellman Problem. In order to prove the security of MTI/C0, we clearly need to make the assumption that the Computational Diffie-Hellman problem is intractable: given g^x and g^y, it is hard to compute g^{xy} for random elements $x, y \in \mathbb{Z}_p$. In order to deal with active attacks, we also need another computational hardness hypothesis that is an extension of the above CDH:

> **2-out-of-3 Computational Diffie-Hellman**.
>
> Given $X = g^x$ and $Y = g^y$, for random $x, y \in \mathbb{Z}_p$, compute a pair (Z, T) of elements in G, where $Z \neq 1$ and T is the CDH value of X, Y and Z: $T = Z^{xy}$.

First, it is clear that 2-3-CDH is at most as difficult as CDH. Indeed, if an adversary manages to compute $h = g^{xy}$, (g^z, h^z) is a correct 2-3-CDH answer for any choice of $z \in \mathbb{Z}_p^\star$. Moreover, it is not more difficult than the inverse-DH because by setting $Z = g^{1/y}$ where $Y = g^y$, (Z, X) is a correct answer. As a consequence, a tight reduction from 2-3-CDH to CDH would imply a tight reduction from Inv-CDH to CDH.

In a cyclic group of composite order, the probability in breaking 2-3-CDH is not smaller than $1/\omega$, where ω is the size of the smallest non-trivial subgroup of G. Indeed, an attacker can always choose at random two elements (Z, T) of order ω and then, since the order of $T' = \mathsf{CDH}(X, Y, Z)$ divides ω, and since there is only one subgroup of order ω in the cyclic group G, $T = T'$ with probability $1/\omega$.

In groups of prime order where the discrete logarithm is hard, which our analysis focuses on, it seems reasonable to expect that no adversary can break 2-3-CDH in polynomial time and with a non-negligible probability. Let us denote by $\mathsf{Succ}_{\mathsf{CDH}}(t)$ and $\mathsf{Succ}_{\text{2-3-CDH}}(t)$, for the maximum winning probability of an attacker running in time t against CDH and 2-out-of-3 Computational Diffie-Hellman in G, respectively. The probability is averaged over all possible challenges (X, Y) and over the randomness of the attacker.

2-3-CDH and Active Attacks. In the next section, we show that the intractability of 2-3-CDH is enough to guarantee the security of MTI/C0. Conversely, solving 2-3-CDH does not seem to be enough for an attacker to impersonate a user in a MTI/C0 session.

3.2 MQV

In the specification of MQV, we have a function f from $G \to \mathbb{Z}_p$. In the actual description of MQV [20, 27], G is a prime order subgroup of an elliptic curve group over a finite field $\mathbb{F}_q$, where q is a n-bit prime; for $P = (x, y) \in G$, $x, y \in [0, q-1]$, $f(P) = x \bmod 2^{\lceil n/2 \rceil} + 2^{\lceil n/2 \rceil}$.

In the following, we use the multiplicative notation for the group G.

The private key s_u of a user U is a random element in $\mathbb{Z}_p$ and the corresponding public key equals $K_u = g^{s_u}$.

Key Material Agreement. A (resp. B) draws a random element r_a (resp. r_b) in $\mathbb{Z}_p$. Then A computes $R_a = g^{r_a}$ and sends it to B. Similarly, B computes $R_b = g^{r_b}$ and sends it to A. The key material KM is then computed by each person according to the relation

$$\begin{aligned}\mathsf{KM} = g^{(r_a + f(g^{r_a})s_a)(r_b + f(g^{r_b})s_b)} &= \left(R_a \times {K_a}^{f(R_a)}\right)^{(r_b + f(R_b)s_b)} \\ &= \left(R_b \times {K_b}^{f(R_b)}\right)^{(r_a + f(R_a)s_a)}.\end{aligned}$$

Therefore $$\varphi(r_a, M_A = (s_a, K_a, K_b)) = g^{r_a} = R_a$$

and $$\psi(R_a, r_b, M_B = (s_b, K_a, K_b)) = \left(R_a \times {K_a}^{f(R_a)}\right)^{(r_b + f(g^{r_b})s_b)}.$$

***f*-Randomized Computational Diffie-Hellman Problem.** As for MTI/C0, we need a new assumption, derived from CDH, for proving the security of MQV. It depends on the function f, hence the notation f-RCDH. As shown below, f-RCDH must be hard for MQV to withstand active attacks. We also show in section 5.1 that the intractability of RCDH can be reduced to the one of CDH under some additional assumptions on f.

> **f-RCDH**
>
> Given $X = g^x$ and $Y = g^y$, for randomly chosen $x, y \in \mathbb{Z}_p$, find $R, Z \in G$ such that $Z = R^x \times g^{f(R)\,x\,y}$.

With $r = \log_g R$ (which the attacker does not need to know), the above relation rewrites $Z = g^{x(r + f(R)y)}$.

As for any computational problem, $\mathsf{Succ}_{f\text{-RCDH}}(t)$ is the maximum winning probability of an adversary running in time t against f-RCDH in G, averaged over X, Y and the random tape of the adversary.

Note that f-RCDH is not more difficult than CDH, because knowing $h = g^{xy} = \mathsf{CDH}(X, Y)$, one can answer a valid pair (Z, R), by choosing $R = g^r$ and $Z = X^r h^{f(R)}$.

***f*-RCDH and Active Attacks.** Solving f-RCDH allows to impersonate the responder (denoted by B in our description) in a MQV session: given the public keys K_A, K_B of A and B and the random value R_A sent by A, B can be impersonated to A using a correct f-RCDH answer $(R_\mathsf{B}, \mathsf{KM})$ to the challenge $(X = R_\mathsf{A} \times K_\mathsf{A}^{f(R_\mathsf{A})}, Y = K_\mathsf{B})$. Indeed, if R_B is used as the random value sent to A, then the resulting key material is KM. Note that no random oracle hypothesis is used here.

4 Sketch of Proof

As explained in the introduction, we follow the real-or-random model, as described in appendix A. In this scenario, the attacker E plays against a simulator S and has complete control of the exchanges between user instances. The simulator S draws a random bit b at the beginning of the game and E's goal is to guess this bit b. The attacker E can perform `Test` and `Corrupt` queries to obtain respectively session keys and long-term private keys of users. Before any `Corrupt` query occurs, the answers of `Test` queries depend on b: they are either the real keys (if $b = 1$) or random values (if $b = 0$). In both cases, the answers to two queries asked to partners in a session are the same. After a `Corrupt` query occurs, `Test` queries are answered by the real keys only. After getting long-term private keys, the adversary is indeed able to compute the session keys itself. Furthermore, forward-secrecy only considers the semantic security of keys agreed before any corruption.

Note that `Test` queries can only be asked to users who actually hold a session key, and thus after reception of a correct key confirmation at the end of the protocol run, so that they are "convinced" that they actually share a session key with their intended partner.

The proof is performed with the now classical game technique [26, 25]. The first game is the real game in which we want to upper bound the success probability of E.

First, session flow collisions are ruled out. This is easy, because not all the randomness of the exchanged values in a protocol run is controlled by the attacker: at least one of the two values exchanged at the beginning of a run is properly drawn in G by S, and the collision probability between two sessions is therefore upper-bounded by $1/p$. Informally, in the remaining game executions, session keys are uncorrelated because of the random oracle hypothesis and the inclusion of the session flow in key derivations.

Next, the attacker key confirmations that are correct "by chance", i.e. although the right query was not made to the random oracle, are refused. There are not too many of them if the output size of H, h, is large enough.

Active attacks before `Corrupt` queries are then artificially blocked. This is performed by refusing key confirmations not originating from the simulator.

Because correct key confirmations produced with incorrect oracle inputs are already forbidden, E sees the difference between this new game and the previous one only if it manages to produce a correct oracle input for a key confirmation. To show that this happens with negligible probability, an instance of a custom problem is introduced in the public keys of two users, such that the oracle input corresponding to a key confirmation is the answer to this challenge.

After this crucial step, we are in a game where no active attack can be performed in sessions before `Corrupt` queries. A CDH challenge is finally introduced in one of these sessions; key confirmations and the final key are simulated by random values. Again because of the random oracle hypothesis, E has to solve the CDH problem to be able to ask a relevant question to the oracle, in order to gain some advantage in guessing b or observing inconsistencies in key confirmations.

We could use the Diffie-Hellman random self-reducibility to introduce a CDH challenge in *all* sessions before a `Corrupt` query, thereby gaining a factor q_s in the security reduction. However, in a concurrent model, many sessions can be "pending" when the first `Corrupt` query occurs; these sessions require a special simulation. The simulator

would therefore have to guess correctly the set of pending sessions, leading to a loss factor 2^{q_s}. This is why the challenge is only introduced in one session.

For simplicity, we limited the scope of the model and the proof (appendices A and B) to a two-user setting. However, since the identities of both parties are included in the session flows and in all key derivations, the generalization of the proof to a n-user setting is straightforward.

Finally, we prove that E's advantage in distinguishing real keys from random ones within time t in a prime-order group G having p elements is bounded by

$$2q_H \times (\mathsf{Succ}_{\mathsf{P}}(t, G) + q_s \mathsf{Succ}_{\mathsf{CDH}}(t, G)) + \frac{q_s^2}{p} + q_s 2^{-h}.$$

where $\mathsf{Succ}_{\mathsf{P}} = \mathsf{Succ}_{\text{2-3-CDH}}$ for MTI/C0 and $\mathsf{Succ}_{\mathsf{P}} = \mathsf{Succ}_{f\text{-RCDH}}$ for MQV.

5 Intractability Results

5.1 f-RCDH and CDH are Equivalent for a Random Oracle f

In this section, we prove that if the function f of f-RCDH can be modeled by a random oracle, f-RCDH is equivalent to CDH. We already know that f-RCDH reduces to CDH without any special assumption (see section 3.2). For the converse implication, we suppose E is an attacker against f-RCDH that has advantage $\mathsf{Adv}(t, q_f)$, where q_f is the number of E's f-queries. Given a CDH challenge (X, Y), we get it as a f-RCDH challenge and assume that E returns (R, Z) such that

$$Z = \mathsf{CDH}(X, R \times Y^{f(R)}) = \mathsf{CDH}(X, Y)^{f(R)} \times \mathsf{CDH}(X, R).$$

Then we can replay part of that successful run and change the function f at the crucial query R to induce the attacker into producing another correct answer (Z', R) to the challenge, with a different value $f'(R)$. Then $Z'/Z = \mathsf{CDH}(X, Y)^{f'(R)-f(R)}$, which easily leads to $\mathsf{CDH}(X, Y)$.

To compute a lower-bound for the success probability of this technique, we need the following splitting lemma [22]:

Lemma 1 (Splitting Lemma) *Let P a probability on a product space $X \times Y$ and $Q \subset X \times Y$.*

Define $$Q' = \left\{ (x, y) \in Q \;\middle|\; \underset{y' \in Y}{P}[(x, y') \in A] \geq P[Q]/2 \right\}$$

Then $$P[Q'|Q] \geq 1/2$$

Let p_c be the collision probability of f and p_{max} be the guessing probability of $f \bmod p$, i.e. the maximum probability of any output value of $f \mod p$, with $\#G = p$. If the output of f is a random uniform h-bit string with $2^h < p$, $p_c = p_{max} = 2^{-h}$.

We suppose without loss of generality that each query is submitted at most once by the attacker. With probability less than $2/p$, X or Y is equal to 1. In the other cases, $\mathsf{CDH}(X, Y)$ is a generator of G. Then, if R is not among the f-queries submitted by the attacker, its probability of success is bounded by p_{max} because of the term $\mathsf{CDH}(X, Y)^{f(R)}$ in the f-RCDH relation. Overall, with probability $\mathsf{Adv}' \geq \mathsf{Adv} - 2/p - p_{max}$, the attacker produces a correct output (R, Z) and makes the query $f(R)$. Now,

let Q_i be the event "E produces a correct output (Z, R), the i-th f-query of E being $f(R)$". Let Adv'_i be the probability of Q_i. Then

$$\sum_{i \leq q_f} \mathsf{Adv}'_i = \mathsf{Adv}'.$$

Let us fix i. The whole behavior of the attacker only depends on its random tape and on the oracle answers. Let us split these inputs into the ones occurring before the i^{th} oracle answer ($x \in X$) and the ones after and including that answer ($y \in Y$). Let us now apply the splitting lemma 1 with $Q = Q_i$. It states that, given $u = (x, y) \in_R Q_i$, with probability 1/2, we have

$$\mathsf{P}[Q_{i,x}] \geq \mathsf{Adv}'_i/2 \quad \text{with} \quad Q_{i,x} = \{y' \in Y | (x, y') \in Q_i\}.$$

Therefore, we can perform two executions of E as follows. The first execution is random. With probability greater than Adv', it yields a correct answer (R, Z), and $f(R)$ is queried on some query of index i. Let x (resp. y) the inputs of E before (resp. after) the i^{th} query. We run again the same execution with inputs x before query i, but y' after query i. With probability 1/2, inputs x of the attacker before query i are such that $\mathsf{P}[Q_{i,x}] \geq \mathsf{Adv}'_i/2$. In that case, with probability $\mathsf{Adv}'_i/2$, E produces again a correct output (Z', R') and $f(R')$ is queried on query i. Since inputs before query i are equal in both executions, $R = R'$. Finally, except with probability $1 - p_c$, answers f_1 and f_2 for $f(R)$ are different in both executions. If all these conditions are met, $\mathsf{CDH}(X, Y)$ can be easily extracted. Overall, since the probability to be in case i is $q_i = \mathsf{Adv}'_i/\mathsf{Adv}'$, the attacker breaks CDH with probability

$$\mathsf{Succ}_{\mathsf{CDH}} \geq \mathsf{Adv}'/2 \sum_i \left[q_i \, \mathsf{Adv}'_i/2\right] - p_c = 1/4 \sum_i \left[\mathsf{Adv}'^2_i\right] - p_c \geq \frac{\mathsf{Adv}'^2}{4q_f} - p_c$$

because of the Cauchy-Schwarz inequality. Finally, if E runs in time t the attacker against CDH runs in time $2t$ and succeeds with probability not less than

$$\mathsf{Succ}_{\mathsf{CDH}}(2t) \geq \frac{[\mathsf{Succ}_{f\text{-}\mathsf{RCDH}}(t, q_f) - 2/p - p_{max}]^2}{4q_f} - p_c.$$

This proof of equivalence between f-RCDH and CDH shows that replacing the function f of MQV by a cryptographic hash function can improve the security of MQV, while not much impairing its performance. This is a case for HMQV [12], where each term $f(R)s_{\mathsf{A}} + R$ in the key material is replaced by $H(R||\mathsf{B})s_{\mathsf{A}} + R$ with H a hash function modeled by a random oracle.

5.2 Generic Group Model

Generic groups were introduced in [24]: a *generic group* is a group $(G, +)$ whose elements are represented randomly. Thus an algorithm E working in a generic group G does not perform group computations itself, but rather makes queries to oracles that answer with representations, in some set I, of the results. Two representations are equal if and only if the corresponding elements are equal. In the sequel, $G = \mathbb{Z}_p$ and $I = [0, p-1]$. Through the group oracle, E can multiply existing elements, and introduce new random elements.

Elements of G represent logarithms, and the representation of some x corresponds to g^x. In a generic group, nothing can be learned from g^x, except log equality: if $g^x = g^y$, $x = y$.

We define a *Generic group problem* that enables to study variants of the computational Diffie-Hellman problem in that model. An adversary E plays against a generic group. Some multivariate polynomial $\varphi(X_1, \ldots, X_k, Y_1, \ldots, Y_\ell)$ is fixed. Some coefficients of φ might depend in an arbitrary way of E's behavior. For values of $x_1, \ldots, x_k$ chosen by the simulator, and knowing $g^{x_1}, \ldots, g^{x_k}$, the goal of E is to compute $Y_1 = g^{y_1}, \ldots, Y_\ell = g^{y_\ell}$ such that $\varphi(x_1, \ldots, x_k, y_1, \ldots, y_\ell) = 0$.

All elements manipulated by E are linear polynomials in $x_1, \ldots, x_k$ and some new random elements $x_{k+1}, \ldots$ introduced through the group oracle. Let us call P_i the polynomial corresponding to y_i. P_i is a random variable. Then we have the following

Theorem 1 *Let $d = \deg(\varphi)$ and $\mathsf{P_m}$ be an upper bound for the probability*

$$\mathsf{P_m} = \mathsf{P}[\varphi(X_1, \ldots, X_k, P_1(X_1, \ldots, X_k), \ldots, P_\ell(X_1, \ldots, X_k)) = 0]$$

Then the probability that E wins after q_G queries satisfies

$$\mathsf{Succ}(q_G) \leq \mathsf{P_m} + \frac{(3q_G + k + 2)^2}{2p} + \frac{d}{p}$$

For example, the plain CDH problem corresponds to $\varphi(x_1, x_2, y_1) = x_1 x_2 - y_1$; in that case, $\mathsf{P_m} = 0$ because for any linear expression y_1 in x_1 and x_2, and possibly other variables, $\varphi \neq 0$.

Proof

We define a game corresponding to the challenge of E.

Generic Group Game 0. A simulator S chooses $x_1, \ldots, x_k$ randomly in G^k, outputs the corresponding representations $r_1, \ldots, r_k$ to E. E has access to an oracle σ that, on input $(a, b, r, r') \in \mathbb{Z}^2 \times I^2$, answers with the representation of $ax + bx'$, where r is the representation of x and r' the representation of x'. The connection between representations and elements of G is managed by the simulator through a list $\mathcal{L}$ of pairs (x, r) associating an element with its representation. A representation r in a σ-query input does not need to correspond to an element of G in $\mathcal{L}$; if it does, the corresponding element is used, otherwise a random element x is drawn by the simulator in G and bound to r, that is, (x, r) is added to $\mathcal{L}$. The same rule applies for the answer to the query: if $ax + bx' = x''$ with $(x'', r'') \in \mathcal{L}$, r'' is answered. Otherwise, a random representation r'', not yet bound to any element of G, is chosen in G, (x'', r'') is added to $\mathcal{L}$, and the answer to the σ query is r''. Overall, each σ-query adds at most 3 pairs to $\mathcal{L}$.

Initially, $\mathcal{L} = \{(0, r_z), (1, r_e), (x_1, r_1), \ldots, (x_k, r_k)\}$; E is given $r_z, r_e, r_1, \ldots, r_k$. E's goal is to output $r'_1, \ldots, r'_\ell$ corresponding to $y_1, \ldots, y_\ell$ in G that, together with the x_i's, cancel φ. The last ℓ queries of E are assumed to be of the form $\sigma(1, 0, r'_i, \_)$. E has won if $\varphi(x_1, \ldots, x_k, y_1, \ldots, y_\ell) = 0$ where $(y_i, r'_i) \in \mathcal{L}$.

Generic Group Game 1. In Game 1, random values in G are replaced by unknowns X_i. Representations of elements correspond to linear combinations of these unknowns with coefficients in $\mathbb{Z}_p$, or polynomials in $\mathbb{Z}_p[X_1, \ldots, X_n, \ldots]$, as follows.

Initially, $\mathcal{L} = \{(0, r_z), (1, r_e), (X_1, r_1), \ldots, (X_k, r_k)\}$; E is given $r_z, r_e, r_1, \ldots, r_k$. When E performs a σ-query using a representation r not yet bound to any element of G, instead of choosing a new random element in G, the simulator introduces a new unknown X_i. In a query (a, b, r, r'), if r represents a polynomial F and r' represents F' (F and F' are either new unknowns or polynomials coming from $\mathcal{L}$), the simulator first computes $F'' = aF + bF'$. As before, if (F'', r'') is in $\mathcal{L}$, r'' is answered, and otherwise a random representation is chosen among the ones not yet appearing in $\mathcal{L}$. All polynomials in $\mathcal{L}$ are affine.

Before stopping the game, E outputs $r'_1, \ldots, r'_\ell$ through σ-queries as in game 0. E wins if

$$\varphi(X_1, \ldots, X_k, P_1, \ldots, P_\ell) = 0$$

where $(P_i, r'_i) \in \mathcal{L}$.

Difference between E's success probabilities in game 0 and game 1. In game 1, representations of different polynomials P_1, P_2 always differ, while in game 0 they differ if and only if $P_1(x_1, \ldots, x_k) \neq P_2(x_1, \ldots, x_k)$.

Let $F_1 = 0, F_2 = 1, F_3 = X_1, \ldots, F_n$ be the polynomials of $\mathcal{L}$ at the end of the game: n is bounded by $3q_G + k + 2$. Note that $\Delta_{i,j} = F_i - F_j \neq 0$ for $i \neq j$. We need the following lemma from [23]:

Lemma 2 *Let p be a prime and F a m-variable polynomial with coefficients in $\mathbb{Z}_p$, of total degree d. Then the probability that a random value of $\mathbb{Z}_p^m$ is a root of R is at most d/p.*

The probability that one of the $\Delta_{i,j}$ cancels at some specific value $\mathbf{x} = (x_1, \ldots, x_k)$ is therefore bounded by $n^2/2p$.

Assuming no $\Delta_{i,j}$ cancels in $\mathbf{x}$, game 2 perfectly simulates game 1. However, the success criterion in game 2 is stricter than in game 1. The probability that $\psi(X_1, \ldots, X_k) = \varphi(X_1, \ldots, X_k, P_1, \ldots, P_\ell) \neq 0$ but $\psi(\mathbf{x}) = 0$ for the polynomials $P_1 = F_{i_1} \ldots, P_\ell = F_{i_\ell}$ chosen by E among $\mathcal{L}$ is bounded by d/p. Indeed, if $\psi \neq 0$, it is of degree $\leq d$ because the P_i are linear, and vanishes in $\mathbf{x}$ with probability $\leq d/p$.

Overall,

$$|\mathsf{Succ}_1 - \mathsf{Succ}_0| \leq \frac{(3q_G + k + 2)^2}{2p} + \frac{d}{p}.$$

Finally, in game 1, E wins if and only if $\varphi(X_1, \ldots, X_k, P_1, \ldots, P_\ell) = 0$. This happens with probability $\mathsf{P_m}$.

□

5.3 2-3-CDH and the Generic Group Model

Our 2-3-CDH problem corresponds to the polynomial $\varphi(x_1, x_2, y_1, y_2) = x_1 x_2 y_1 - y_2$. Indeed, the answer (Z, T) to the 2-3-CDH challenge $(X = g^x, Y = g^y)$ is supposed to satisfy $T = Z^{xy}$. This is equivalent to the above equation provided $X = g^{x_1}$, $Y = g^{x_2}$, $Z = g^{y_1}$ and $T = g^{y_2}$. Since $Z \neq 1$, a valid answer of the adversary is such that y_1 is a non-zero affine polynomial and $\varphi \neq 0$, therefore $\mathsf{P_m} = 0$. Therefore, using $k = 2$ and $d = 2$, theorem 1 yields

$$\mathsf{Succ}_{\text{2-3-CDH}}(q_G) \leq \frac{(3q_G + 4)^2}{2p} + \frac{2}{p} = 9 \times \frac{q_G^2}{2p} + 12 \times \frac{q_G}{p} + \frac{10}{p}.$$

5.4 f-RCDH and the Generic Group Model

For our f-RCDH problem, $\varphi(x_1, x_2, y_1, y_2) = x_1(f(r)x_2 + y_1) - y_2$, where r is the representation of y_1. This case is a little bit more complicated than for 2-3-CDH, because φ depends on E's answers through the term $f(r)$.

Let $\psi(X_1, X_2, \ldots) = X_1(f(r)X_2 + P_1(X_1, X_2, \ldots)) - P_2(X_1, X_2, \ldots)$ where the P_i are the polynomial representations of the Y_i. Unknowns X_i for $i > 2$ represent random values in the group introduced by E. We want an upper bound on $\mathsf{P}[\psi = 0]$ at the end of the game.

We know that $\deg(\psi) \geq 2$ as soon as $f(r)X_2 + P_1$ is not constant. Either r was chosen by the adversary and $P_1 = X_i$ with $i > 2$, or P_1 is an affine polynomial chosen by the adversary through some sequence of computations and r is random. In the first case $f(r)X_2 + P_1$ is not constant. In the second case, r is a random uniform value in $I \backslash I'$, where $I' = \{r_1, \ldots, r_n\}$ and the r_i are the other representations already in $\mathcal{L}$ at the time of the query producing r. The best E can do is to set P_1 to $-u\,X_2$ where u is the most likely output of f for a random input x in $I \backslash I'$.

Let $p_{max}(I') = \max_{i \in I \backslash I'} \mathsf{P}[f(x) = i | f(x) \notin I']$: $f(r)X_2 + P_1$ is constant with probability less than $p_{max}(I')$. Overall if $p_{max}(n)$ is a uniform bound over I' of $p_{max}(I')$ for $\#I' \leq n$, $\mathsf{P_m} \leq n p_{max}(n)$ and theorem 1 yields with $n = 3q_G + 4$

$$\mathsf{Succ}_{f\text{-RCDH}}(q_G) \leq n\, p_{max}(n) + \frac{n^2}{2p} + \frac{2}{p}$$

Discussions about the Maximum Probability p_{max}. In the specification of MQV [27], f is the truncation of the $\ell = \lfloor \log_2(p)/2 \rfloor + 1$ LSBs of its input. Let $\alpha = \lceil p/2^\ell \rceil$. Then every element in $[0, 2^{\ell-1}]$ is the image of at most α elements in I, therefore $p_{max}(0) \leq \alpha/p$. If n elements are removed in I, $p_{max}(n) \leq \alpha/(p-n)$, and therefore if $n \leq p/2$,

$$p_{max}(n) \leq \frac{p/2^\ell + 1}{p - n} \leq 2\,\frac{p/2^\ell + 1}{p} = 2\left(2^{-\ell} + \frac{1}{p}\right) \leq 2\left(\frac{1}{\sqrt{p}} + \frac{1}{p}\right).$$

Overall, with MQV, the winning probability of E against f-RCDH satisfies

$$\mathsf{Succ}_{f\text{-RCDH}}(q_G) \leq \frac{9\,q_G^2}{2p} + \frac{18\,q_G}{p} + \frac{18}{p} + \frac{6\,q_G}{\sqrt{p}} + \frac{8}{\sqrt{p}}$$

As long as $n = 3\,q_G + 4 \leq p/2$, this last hypothesis being perfectly sensible for cryptographic purposes.

6 Key Exchange Implementation Choices

Our security proof highlights the importance of several implementation choices when working with Diffie-Hellman-like key exchange algorithms:

- **Work in a prime order group G.** In our case, the computational problem related to MTI/C0, 2-3-CDH, has a security that depends on the size of the smallest non-trivial subgroup of G. As for f-RCDH, if G has non-trivial subgroups, trade-offs can be devised to force the common key to belong to some subgroup of G; the proof of hardness of f-RCDH in a composite-order generic group would yield a bound depending on the size of the largest prime order subgroup of G.

– **Use the session flow, including parties identities, to derive keys.** This "freezes" active attacks by de-correlating keys between users and sessions. Note that this is not specific to the signature-less case: an unknown key-share attack can be devised against STS because it does not follow this principle [10]. Including the user identities is of course crucial in a setting with more than two users.
– **Confirm the keys.** Without key confirmations, an adversary against a signature-less protocols can impersonate a user during the key negotiation, and then wait for a long-term key leakage to compute the session key. On the contrary, a key confirmation prevents the other party to output material enciphered with the session key before it is sure that its partner actually knows the key.

Acknowledgements

The authors were supported in part by the European Commission through the IST Program under Contract IST-2002-507932 ECRYPT.

References

1. M. Abdalla, O. Chevassut, and D. Pointcheval. One-time Verifier-based Encrypted Key Exchange. In S. Vaudenay, editor, *Public Key Cryptography*, volume 3386 of *LNCS*, pages 47–74. Springer-Verlag, 2005.
2. M. Abdalla, P.-A. Fouque, and D. Pointcheval. Password-Based Authenticated Key Exchange in the Three-Party Setting. In S. Vaudenay, editor, *Public Key Cryptography*, volume 3386 of *LNCS*, pages 65–84. Springer-Verlag, 2005.
3. M. Bellare, R. Canetti, and H. Krawczyk. A modular Approach to the design and Analysis of Authentication and Key Exchange Protocols (extended abstract). In *STOC '98*, pages 419–428. ACM Press, 1998.
4. M. Bellare, A. Desai, E. Jokipii, and P. Rogaway. A Concrete Security Treatment of Symmetric Encryption: Analysis of the DES Modes of operation. In *Proceedings of the 38th Symposium of Foundations of Computer Science*, pages 394 – 403. IEEE Computer Security Press, 1997.
5. M. Bellare, D. Pointcheval, and P. Rogaway. Authenticated Key Exchange Secure Against Dictionary Attacks. In B. Preneel, editor, *Advances in Cryptology – Eurocrypt 2000*, volume 1807 of *LNCS*, pages 470–484. Springer-Verlag, 2000.
6. M. Bellare and P. Rogaway. Random Oracles are Practical: A Paradigm for Designing Efficient Protocols. In *Proceedings of the 1st ACM Conference on Computer and Communications Security*, pages 62 – 73. ACM Press, 1993.
7. M. Bellare and P. Rogaway. Entity Authentication and Key Distribution. In *Advances in Cryptology – Crypto '93*, volume 773 of *LNCS*, pages 232–249. Springer-Verlag, 1994.
8. S. Blake-Wilson, D. Johnson, and A. Menezes. Key Agreement Protocols and their Security Analysis. In *Cryptography and Coding*, volume 1355 of *LNCS*, pages 30–45. Springer Verlag, 1997.
9. S. Blake-Wilson and A. Menezes. Authenticated Diffie-Hellman Key Agreement Protocols. In *Selected Areas in Cryptography*, pages 339–361, 1998.
10. S. Blake-Wilson and A. Menezes. Unknown Key-Share Attacks on the Station-to-Station (STS) Protocol. In *Public Key Cryptography*, pages 154–170, 1999.
11. E. Bresson, O. Chevassut, D. Pointcheval, and J.-J. Quisquater. Provably Authenticated Group Diffie-Hellman Key Exchange. In *ACM Conference on Computer and Communications Security*, pages 255–264. ACM Press, 2001.
12. R. Canetti and H. Krawczyk. Analysis of Key-Exchange Protocols and Their Use for Building Secure Channels. In *Advances in Cryptology – Eurocrypt'01*, volume 2045 of *LNCS*, pages 453–474, London, UK, 2001. Springer-Verlag.
13. W. Diffie and M. E. Hellman. New Directions in Cryptography. *IEEE Transactions on Information Theory*, 1976.
14. W. Diffie, P. van Oorschot, and M. Wiener. Authentication and Authenticated Key Exchanges. *Design, Codes and Cryptography*, 2(2):107–125, 1992.

15. I. R. Jeong, J. Katz, and D. H. Lee. One-Round Protocols for Two-Party Authenticated Key Exchange. In *Applied Cryptography and Network Security 2004 Proceedings*, volume 3089 of *Lecture Notes in Computer Science*, pages 220–232. Springer, 2004.
16. B. S. Kaliski Jr. An Unknown Key-share Attack on the MQV Key Agreement Protocol. *ACM Trans. Inf. Syst. Secur.*, 4(3):275–288, 2001.
17. H. Krawczyk. HMQV: A High-Performance Diffie-Hellman Protocol. In Victor Shoup, editor, *Proceedings of CRYPTO 2005*, volume 3621 of *LNCS*, pages 546–566. Springer-Verlag, August 2005.
18. K. Lauter and A. Mityagin. Security Analysis of KEA Authenticated Key Exchange. Cryptology ePrint archive, Report 2005/265, available at `http://eprint.iacr.org`.
19. T. Matsumoto, Y. Takashima, and H. Imai. On Seeking Smart Public-key Distribution Systems. *Transactions of the IECE of Japan*, E69:99–106, 1986.
20. A. Menezes, M. Qu, and S. Vanstone. Some New Key Agreement Protocols Providing Mutual Implicit Authentication. *Workshop on Selected Areas in Cryptography (SAC '95)*, pages 22–32, 1995.
21. A. Menezes, P. van Oorschot, and S. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1996.
22. D. Pointcheval and J. Stern. Security Arguments for Digital Signatures and Blind Signatures. *Journal of Cryptology*, 13(3):361–396, 2000.
23. J. T. Schwartz. Fast Probabilistic Algorithms for Verification of Polynomial Identities. *J. ACM*, 27(4):701–717, 1980.
24. V. Shoup. Lower Bounds for Discrete Logarithms and Related Problems. In W. Fumy, editor, *Advances in Cryptology – Eurocrypt 2000*, volume 1233 of *LNCS*, pages 256–266. Springer Verlag, 1997.
25. V. Shoup. A proposal for an ISO standard for public key encryption, 2001. Cryptology ePrint report 2001/112.
26. V. Shoup. OAEP reconsidered (Extended Abstract). In J. Kilian, editor, *Advances in Cryptology – Crypto'01*, volume 2139 of *LNCS*, pages 239 – 259. Springer-Verlag, 2001.
27. Standard for Efficient Cryptography Website. `http://www.secg.org/`.

A Security Model

The security requirements are formalized in a Real-or-Random game between an attacker E and a simulator S simulating two users A and B. H is modeled by a random oracle. However, we *do not use the random oracle programmability*; therefore it is only assumed that H is "black-box". H can be seen as a random oracle that is outside the attacker *but also outside the simulator*. Each time E "presses the button" to get a hash value of some message, S gets the input message together with the hash value chosen by the oracle. The image space of H is $\mathcal{K}$, the key space.

At the beginning of the game between S and E, S draws a random bit b uniformly; b decides whether random values or actual keys will be shown to E. The goal of E is to correctly guess the value of b.

Simulation and Attacker's Queries. E can issue the following queries to S to control sessions and messages exchanged by A and B:

- j = `Initiate` : initiate a new session. The attacker receives a string that is a session ID used in `Test` and `Send` queries.
- `Send`(U, M, j): send message M to user U for session j.

Messages that are supposed to be sent by A or B in the real protocol are actually given by S to the attacker E, along with the index of the session which the messages belong to.

Additionally, E can perform the following queries:

- `Test`(U, j): obtain the session key negotiated after session j from user U (U = A or B);
- `Corrupt`(U): obtain the long-term private key of U = A or B.

Each of the `Send`, `Test`, and `Corrupt` queries models a different attack: `Send` queries allow E to perform Man-in-the-Middle attacks by altering, deleting or inserting messages between A and B, `Test` queries model session key material leakage, while `Corrupt` queries model long-term key material leakage.

A `Test`(U, j) query is answered as follows: if U did not accept the key negotiation of session j (see "Accepted keys" below), an error is returned. Otherwise, U has computed a key k and E gets the following answer:

- if a `Corrupt` query was issued before the `Test` query, the key k is returned;
- if no `Corrupt` query has been issued so far, the answer depends on b: if $b = 1$, k is returned; if $b = 0$, a random value $H'(\mathsf{Flow}(\mathsf{U}, i)) \in \mathcal{K}$ is returned, for some private random oracle H' simulated by the simulator.

Notice that the answer to `Corrupt` queries in the random case ($b = 0$) does not depend on the user on which the query was performed if A and B are partners for the session, as in the real case.

E wins the game if at some point it outputs its answer b' with $b' = b$. E's *advantage* is then

$$\mathsf{Adv} = \left| \mathsf{P}_{b=1}[b' = 1] - \mathsf{P}_{b=0}[b' = 1] \right| .$$

We are looking for an upper bound for Adv, depending on its running time t, its number of H-queries q_H and its number of `Initiate` queries q_s.

Accepted keys. Suppose A and B negotiate a key. If the agreement succeeds, at some point, A will start using the key, which might then leak (this is modeled by `Test` queries). A should not use the key before it is convinced that it actually shares the key with B and B only. To capture this notion, we say that A (or B) *accepts* (the key negotiation) when it is convinced of the authenticity of the computed key, and authorize `Test` query only on accepted keys.

B Security Proof

B.1 Overview

We name the real game between the attacker and the simulator Game 0. Game 1 is equal to game 0 except that the simulator aborts as soon as it observes two session flows in two different sessions that are identical. We then study a derived game 1' that allows us to prove that before a `Corrupt` query occurs, E is unable to perform an active attack. We can therefore replace game 1 with game 3, which is identical to game 1 except that key confirmations on sessions tampered with by the attacker are refused regardless of their actual correctness.

Knowing that we do not have to deal with active attacks before `Corrupt` queries, we are then able to build a Game 4 that connects E's advantage to the difficulty of the CDH problem.

B.2 The Original Game 0

Game Initialization. S generates uniformly at random a bit b, two long-term private keys s_a and s_b and the corresponding public keys.

Session Simulation. Sessions are simulated according to the protocols described in section 3. Since H is modeled by a Random Oracle, besides the queries described in the appendix A, E can perform H-queries. `Test` queries on keys accepted before any `Corrupt` query occurs are answered by

$$H(\mathsf{KM}||0||\mathsf{Flow}(\mathsf{U})) \text{ if } b = 1 \quad \text{and} \quad H'(0||\mathsf{Flow}(\mathsf{U})) \text{ if } b = 0$$

for some private random oracle H'. However, a `Test` query on a key accepted after a `Corrupt` query is always answered with the "real" response $H(\mathsf{KM}||0||\mathsf{Flow}(\mathsf{U}))$. `Corrupt` queries are answered by the corresponding long-term private key.

At this stage, since collisions may happen on outputs of H, a key confirmation might be accepted even if the correct query was not submitted by the attacker to the oracle.

Advantage in Game 0. in Game 0 the advantage of E is $|\mathsf{Adv}_0|$ where

$$\mathsf{Adv}_0 = \mathsf{P}_{b=1}[b' = 1] - \mathsf{P}_{b=0}[b' = 1].$$

B.3 Game 1: Ruling Out Colliding Sessions

Game 1 is equal to Game 0 except if $\mathsf{Flow}(\mathsf{U}, i) = \mathsf{Flow}(\mathsf{V}, j)$ for some $i < j$ (one can have $\mathsf{U} = \mathsf{V}$ or $\mathsf{U} \neq \mathsf{V}$). In that case, S aborts the simulation.

Since the session flow seen from some user U contains at least one random value computed by S, the one emitted by U, if a flow collision occurs, S must have generated two equal random values in G. For one session pair $\{i, j\}$, this happens with probability $1/\#G$. During q_s sessions, this happens with probability $\alpha \leq q_s^2/2p$. Therefore Game 1 is indistinguishable from Game 0 with probability greater that $1 - \alpha$. Considering that E wins if S aborts because of a collision, one has for some $-1 \leq \varepsilon_1 \leq 1$,

$$\mathsf{Adv}_1 = (1 - \alpha)\mathsf{Adv}_0 + \alpha\,|\varepsilon_1|$$

and

$$|\mathsf{Adv}_1 - \mathsf{Adv}_0| \leq \alpha(|\varepsilon_1| + |\mathsf{Adv}_0|) \leq 2\alpha \leq q_s^2/p$$

B.4 Game 2: Demanding the Correct Oracle Query for Key Confirmations

Assume the adversary produces a key confirmation KC for a session of flow Flow and of master secret KM. Assume also that it did not make the correct H-query $H(I)$ corresponding to KM. There are two possibilities:

- Either $H(I)$ is not fixed yet because the H-query I was never performed by S. In that case, the key confirmation produced is correct with probability 2^{-h}.
- Or the H-query I was performed by the S before. It was necessarily when computing a key confirmation of some previous session. Moreover this previous session is not the current one since the two key confirmations exchanged in a session differ. Since Flow is included in I, this means that two sessions flows collide, but session collisions were blocked in game 1.

In game 2, before S accepts a key confirmation, it checks that a H-query with the correct input was made to generate it; if not, the key confirmation is rejected. The reasoning above shows that advantages of E in game 1 and in game 2 satisfy

$$|\mathsf{Adv}_2 - \mathsf{Adv}_1| \leq q_s 2^{-h}$$

B.5 Game 2': Ruling Out Active Attacks

In this section, we replace game 2 by a game 2' where we arbitrarily forbid active attacks before the first `Corrupt` query of the adversary. This is performed by refusing any key confirmation in sessions that are not fully passive.

We show that except with small probability, the observations of any arbitrary adversary E in game 2' are consistent. Because of this, if we change game 2 into a game 3 where active attacks before the first `Corrupt` query are arbitrarily blocked, no polynomial-time adversary can tell the difference between game 2 and game 3.

One may wonder why we need to introduce two different games to rule out active attacks, game 2' and 3. The game of actual interest is game 3, while game 2' is only studied to show that some event in game 3, the observation by the adversary of an inconsistency in the simulation, cannot happen.

Game 2' Overview. In game 2', the goal of the simulator is to compute an answer to a 2-3-CDH-challenge (for MTI/C0) or an f-RCDH-challenge (for MQV). The input of the challenge is introduced in the public keys of the users $K_{\mathsf{A}} = g^{s_a}$ and $K_{\mathsf{B}} = g^{s_b}$. As a consequence, the simulator cannot answer `Corrupt` queries. Therefore it simply aborts if the adversary outputs a `Corrupt` query[2].

Another consequence is that the simulator is not able to answer consistently to all the queries of the attacker. Moreover, the simulator refuses key confirmations in non-passive sessions: this might yield other inconsistencies in the simulation.

Our goal is to show that the probability of the adversary to perform an observation showing any inconsistency in the simulation is connected to the success probability of the simulator.

Answers to H-queries. H is simulated by a random oracle as in game 2.

Session Simulation. In game 2', a passive session of index i is simulated as follows:

$$\begin{array}{rcl} & \mathsf{A} \qquad \mathsf{B} & \\ & g^x \longrightarrow & \\ & \longleftarrow g^y & \\ R_1 = H_{kc}(\mathsf{ID_A}||\mathsf{Flow(A)}) & \longrightarrow & \\ & \longleftarrow & R_2 = H_{kc}(\mathsf{ID_B}||\mathsf{Flow(B)}) \end{array}$$

where H_{kc} is a random oracle independent from H. In case of an active attack, the session flow is described by one of the two following diagrams (since the simulation is symmetric, it can always be assumed that A is attacked):

$$\begin{array}{rcl} & \mathsf{A} \qquad \mathsf{B} & \\ & g^x \longrightarrow & \\ & \longleftarrow Y & \\ R_1 = H_{kc}(\mathsf{ID_A}||\mathsf{Flow(A)}) & \longrightarrow & \\ & \longleftarrow R' & \end{array} \quad \text{or} \quad \begin{array}{rcl} & \mathsf{A} \qquad \mathsf{B} & \\ & \longleftarrow Y & \\ g^x & \longrightarrow & \\ & \longleftarrow R' & \end{array}$$

[2] If no `Corrupt` query was performed, the game can also end normally when the adversary answers to its challenge; in that case the answer of the attacker is of no interest for the simulator and is discarded.

In the second active case, since the key confirmation output by the adversary is rejected, there is no key confirmation output by the simulator. In both active cases, there is no accepted session key, so no `Test` query can be performed by the adversary on the session.

In the passive and active cases, although the simulator does not generate values according to the protocol, the distribution of these values is the same as in the real game.

Answers to `Test` queries. A `Test` query on user U and session i is answered (on passive sessions only) by $H_t(\mathsf{Flow}(\mathsf{U}, i))$, where H_t is a random oracle independent from H and H_{kc}. Notice that the joint distribution of answers to `Test` queries is correct: they are uniformly distributed, answers corresponding to different sessions are independent (because the hash input includes session flows which are different because flows cannot collide), and answers of both users in the same passive session are equal (because in that case, flows are equal).

Consistency between H-queries, `Test` queries and key confirmations. To be consistent with game 2, key confirmations produced by the simulator in game 2' should satisfy $\mathsf{KC}(\mathsf{U}) = H(S\|1\|\mathsf{U}\|\mathsf{Flow}(\mathsf{U}, i))$ for some user U in some session i, where $S = g^{\frac{s_a s_b}{xy}}$ for MTI/C0 and $S = g^{(f(g^x)s_a+x)(f(g^y)s_b+y)}$ for MQV. Unfortunately, the simulator is not able to compute that value. As a consequence, it is not able to generate consistent key confirmations; the same problem holds for `Test` queries. This is why key confirmations and answers to H-queries are simulated by independent random oracles.

Extracting an answer to the challenge of the simulator. When the game stops, either after a `Corrupt` query or after E's answer, the simulator tries to extract an answer its challenge from H-queries performed by E in the following way.

The formatting of a H-query indicates whether it corresponds to a key confirmation or a `Test` query, and the target session index. Incorrectly formatted queries can be discarded, since they are just random values independent of the rest of the simulation. Consider H-queries corresponding to a key confirmation or a `Test` query on some session i.

Let y denote the g-log of Y. The simulator knows y iff session i is passive. Since the H-query contains a candidate for $S = g^{\frac{xy}{s_a s_b}}$ (MTI/C0) or $S = g^{(f(g^x)s_a+x)(f(g^y)s_b+y)}$ (MQV), the simulator always has access to x, Y and S. It can therefore compute the pair

$$v = \left(g^{\frac{y}{s_a s_b}}, g^y\right) \quad \text{(MTI/C0)} \quad \text{or} \quad v = \left(g^y, g^{s_a(f(g^y)s_b+y)}\right) \quad \text{(MQV)}$$

and v a candidate answer for the challenge $(K_\mathsf{A}, K_\mathsf{B})$ (notice that in the MTI/C0 case, one necessarily has $g^y \neq 1$).

A remark about static attacks. The knowledge of $h = \mathsf{CDH}(K_\mathsf{A}, K_\mathsf{B})$ allows one to impersonate A to B, or B to A, during a MTI/C0 session: suppose E sends h^α to A, and receives r from A. Then the common key r^α can be computed by E. For each users pair (A, B), $\mathsf{CDH}(K_\mathsf{A}, K_\mathsf{B})$ therefore plays the role of a common secret key. However, this does not contradict the security proof since such a behavior of E would allow S to retrieve h; and indeed, h is not easier to compute than an ephemeral CDH challenge. From a practical standpoint, it does not seem easier to extract h from a crypto device than s_a or s_b and this attack model therefore has a little impact.

Answer to the Challenge. The simulator discards irrelevant H-queries, chooses uniformly at random one of the remaining queries, and answers a corresponding guess for the answer.

Success Probability. We are in one of the two following situations:

- either all the H-queries output by E are incorrect. Then game 2' perfectly simulates game 2, because key confirmations output by E would also have been refused in game 2.
- or the simulator breaks its CDH-like challenge with probability $1/q_H$.

If β is the probability that E makes an observation allowing it to distinguish between game 2 and game 2', then $\beta \leq q_H \mathsf{Succ}_{\mathsf{P}}$ where $\mathsf{Succ}_{\mathsf{P}} = \mathsf{Succ}_{\text{2-3-CDH}}$ for MTI/C0 and $\mathsf{Succ}_{\mathsf{P}} = \mathsf{Succ}_{f\text{-RCDH}}$ for MQV. Game 2' is indistinguishable from game 2 with probability greater than $1 - \beta$.

Because of this result, game 3, which is equal to Game 1 except that key confirmations produced by E before the first `Corrupt` query are rejected, can be distinguished from game 1 with probability $\beta \leq q_H \mathsf{Succ}_{\mathsf{P}}$. Therefore, as in section B.3, one can show that

$$|\mathsf{Adv}_3 - \mathsf{Adv}_2| \leq 2q_H \times \mathsf{Succ}_{\mathsf{P}}.$$

Game 4 Overview.

- There is no real challenge for the attacker anymore[3], and $\mathsf{Adv}_4 = 0$.
- The goal of the simulator is to solve a CDH problem instance $X = g^x$, $Y = g^y$. X and Y are introduced in the random elements exchanged in one session completed before a `Corrupt` query occurs (the "challenge session").
- The challenge session is chosen at random at the beginning of the game. If a `Corrupt` query occurs before it ends, the game is aborted.
- In other sessions, everything is perfectly simulated with values known from the simulator as in game 2. For instance, key confirmations output by the simulator and answers to `Test` queries are computed using the public oracle H with the same inputs as in the real case ($b = 1$) of game 2.
- As in game 3, key confirmations on sessions tampered with by the attacker before a `Corrupt` query occurs are rejected.
- In the target session, key confirmations (resp. answers to `Test` queries) on user U, session i, are answered as in game 3 by a private oracle $H_{kc}(\mathsf{U}||\mathsf{Flow}(\mathsf{U}, i))$ (resp $H_t(\mathsf{Flow}(\mathsf{U}, i))$.
- The simulator uses public keys of known g-log (as required in order to answer `Corrupt` queries).

The target session (of index i) is passive and looks like this for MTI/C0:

$$\begin{array}{rl} X^{r/b} & \longrightarrow \\ & \longleftarrow Y^{s/a} \\ R_1 = H_{kc}(\mathsf{ID_A}||\mathsf{Flow}) & \longrightarrow \\ & \longleftarrow R_2 = H_{kc}(\mathsf{ID_B}||\mathsf{Flow}) \end{array}$$

[3] one could choose a bit b privately at random to have a "challenge", and consider that the adversary wins its challenge if its answer b' is equal to b; however, this amounts to say that by construction in game 4, the winning probability of E is 1/2, or its advantage is 0.

for MQV, we would have:

$$\begin{array}{rl} & X^r \longrightarrow \\ & \longleftarrow Y^s \\ R_1 = H_{kc}(\mathsf{ID_A}||\mathsf{Flow}) & \longrightarrow \\ & \longleftarrow R_2 = H_{kc}(\mathsf{ID_B}||\mathsf{Flow}) \end{array}$$

where r and s are uniformly independently distributed random values.
Other sessions can be passive or active. They are simulated as in game 2.

Answer of the Simulator to its Challenge. As for game 2', let q_H be the number of H-queries related to key confirmations or `Test` queries on sessions before a `Corrupt` query. The simulator can derive from any such query a candidate for $\mathsf{CDH}(X, Y)$ (see the passive case in game 2'). Therefore when E gives its answer to the challenge, which is discarded, the simulator chooses one of the q_H H-queries and answers the corresponding candidate for g^{xy}. If E performed no H-query, the simulator simply outputs a random value.

We argue as in B.5 to show that either the simulator has at least one correct candidate for $\mathsf{CDH}(X, Y)$ for one of the sessions before any `Corrupt` query, or the attacker does not distinguish between game 3 and game 4. With probability above $1/q_s$, the session concerned is the target session.

- If no candidate is equal to $\mathsf{CDH}(X, Y)$, the values observed by E are independent from `Test` queries answers and key confirmations both in game 3 and in game 4. Therefore the simulation is perfect.
- If at least one candidate is equal to $\mathsf{CDH}(X, Y)$, the simulator has probability greater than $1/(q_H q_s)$ to output the correct answer to its challenge.

The probability of E to make an observation allowing it to distinguish between game 3 and game 4 is therefore upper-bounded by $q_H q_s \mathsf{Succ}_{\mathsf{CDH}}$ and $\mathsf{Adv}_4 = 0$. Finally,

$$|\mathsf{Adv}_3| \leq 2 q_H q_s \times \mathsf{Succ}_{\mathsf{CDH}}.$$

Putting everything together, one gets

$$|\mathsf{Adv}_0| \leq 2 q_H \times (\mathsf{Succ}_{\mathsf{P}} + q_s \mathsf{Succ}_{\mathsf{CDH}}) + \frac{q_s^2}{p} + q_s 2^{-h}.$$

where $\mathsf{Succ}_{\mathsf{P}} = \mathsf{Succ}_{\text{2-3-CDH}}$ for MTI/C0 and $\mathsf{Succ}_{\mathsf{P}} = \mathsf{Succ}_{f\text{-RCDH}}$ for MQV.